BRL-TR-3021

(2)

AD-B134 771

TECHNICAL REPORT BRL-TR-3021

BRL

ETHER & BIT BOX PROGRAMS: FIREPOWER CONTROL EXPERIMENT, PART 2 OF 12

GEORGE W. HARTWIG, Jr.

JULY 1989

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

89 7 31 098

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT ~~Distribution limited to US Government Agencies and their contractors; administrative/operational use. Other requests for this document must be referred to Director, USA, BRL.~~
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) USABRL-TR-3021	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION US Army Ballistic Research Laboratory 6b. OFFICE SYMBOL *(If applicable)* SLCBR-SE-W	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State, and ZIP Code)* Aberdeen Proving Ground, MD 21005-5066	7b. ADDRESS *(City, State, and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)*	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Ether & Bit Box Programs: Firepower Control Experiment Part 2 of 12

12. PERSONAL AUTHOR(S)
George W. Hartwig, Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Technical	FROM ______ TO ______	~~87-10~~ July 89	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Artillery Control Environment (ACE); Firepower Control Experiment; Ether; Multiple Forward Observer Scenario (MFOSCE); Bit Box; ACE Display (ADIS); Command Post Research Facility (CPXRF); Interprocess Communication. (AW)

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The computer program *Ether* serves as the communications media and control center for the Artillery Control Environment (ACE) simulation programs. *Ether* sets up all of the interprocess communication required for a desired run and by passing information to the appropriate display programs, allows the user to monitor the progress of the unfolding scenario. The experiment controller also has the ability to alter the probability of successful message transmission during the execution of an experiment.

Bb is the Bit Box driver program. This program serves as an interface between the computer port attached to tactical hardware and the *ether* program. The use of this program allows the *ether* program to use a consistant set of protocols for for communicating with both simulation programs and tactical hardware.

Keywords: Fire Control Computers, Tactical Communications; Man Machine Systems, Interfaces, Message Traffic, Data Acquisition.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL George W. Hartwig, Jr.	22b. TELEPHONE *(Include Area Code)* 301-278-6672 22c. OFFICE SYMBOL SLCBR-SE-W

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

PREFACE

This is one of a series of reports that cover the Ballistic Research Laboratory (BRL) Firepower Control Experiment conducted 2 December to 20 December 1985 at the jointly developed BRL & Human Engineering Laboratory (HEL) Command Post Exercise Research Facility (CPXRF). A total of twelve reports will be published and each covers a specific topic.

This report is part 2 of 12; a complete list with the associated authors follows:

TITLE	AUTHORS
Concept and Purpose Firepower Control Experiment Part 1 of 12	Samuel C. Chamberlain
Ether & Bit Box Programs Firepower Control Experiment Part 2 of 12	George W. Hartwig
ADIS Program Firepower Control Experiment Part 3 of 12	Virginia A. Kaste
MFOSCE Program Firepower Control Experiment Part 4 of 12	Louise D. Kokinakis
Software Library for *Ether, ADIS, & MFOSCE Programs* Firepower Control Experiment Part 5 of 12	Virginia A. Kaste, George W. Hartwig,
The Enhanced Fire Direction Simulator Firepower Control Experiment Part 6 of 12	Douglas C. Brodeen and Thomas A. DiGiacinto
FDO Display and DMD Converter Programs Firepower Control Experiment Part 7 of 12	Kenneth G. Smith
Gun Simulator (GUNSIM) Program Firepower Control Experiment Part 8 of 12	Eric G. Heilman
Gun Display Unit Hardware Interface Firepower Control Experiment Part 9 of 12	Dr. Mark D. Kregel
Scenario Development and Tactical Input Data Firepower Control Experiment Part 10 of 12	Wendy A. Winner and Maj. William T. Dougherty
Knowledge Acquisition Survey & Analysis Firepower Control Experiment Part 11 of 12	Maj. William T. Dougherty and Richard C. Kaste
Test Design and Analysis Firepower Control Experiment Part 12 of 12	Wendy A. Winner, Ann M. Brodeen, and Jill H. Smith

TABLE OF CONTENTS

TABLE OF CONTENTS (contd)

ETHER & BIT BOX PROGRAMS:

FIREPOWER CONTROL EXPERIMENT PART 2 OF 12

by

GEORGE W. HARTWIG, JR.

BRL Technical Report

ABSTRACT

The computer program *Ether* serves as the communications media and control center for the Artillery Control Environment (ACE) simulation programs. *Ether* sets up all of the interprocess communication required for a desired run and by passing information to the appropriate display programs, allows the user to monitor the progress of the unfolding scenario. The experiment controller also has the ability to alter the probability of successful message transmission during the execution of an experiment.

Bb is the Bit Box driver program. This program serves as an interface between the computer port attached to tactical hardware and the *ether* program. The use of this program allows the *ether* program to use a consistant set of protocols for for communicating with both simulation programs and tactical hardware.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the contributions of Michael J. Muuss who conceived the original design of *ether* and coded the initial version, and to Howard A. Walter who greatly extended the program over the years. I also wish to express thanks to John Leonard of CSC for writing the initial versions of the manual pages which appear in appendices A and B.

LIST OF ILLUSTRATIONS

I. INTRODUCTION

The Artillery Control Environment (ACE) has proven to be a valuable tool in the evaluation and testing of new equipment and techniques in the Fire Support Control (FSC) portion of artillery systems.

The heart of ACE is the 4.2 BSD UNIX[*] operating system with Ballistic Research Laboratory (BRL) enhancements. ACE permits the real-time, interactive simulation of fire support control for any low-level battlefield slice. This can be done in any of the following combinations: a) soldier operated tactical automatic data processing (ADP) devices can "talk" to the computer via the BRL Bit Box, which is a special modem that interfaces TACFIRE hardware to commercial computers (See Appendix A); b) low cost commercial hardware with the appropriate software can emulate tactical hardware; c) interactive computer programs can simulate and replace live player functions. Fire support components that are not "played" explicitly or inputs that are external to the organization being studied can be simulated with messages entered into the system by specialized interactive programs or by a tactical operator following a script. ACE components are interconnected by the program *ether*, which functions as a multichannel communications medium and has the capability to degrade communications. ACE also has the capability to display and store digital message traffic. Experimenters can monitor message traffic in real time which provides for quicker response when controlling tests. The automated collection of digital data, as opposed to manually recorded data, enables faster turn-around time for data reduction and analysis.[1]

Figure 1 is the ACE software configuration developed for the Firepower Control Experiment held in December 1985. Parallelograms depict input and output files, rectangles illustrate programs, ellipses represent terminal displays and beveled rectangles are hardware. On HEL-ACE (a Digital Equipment Corporation Vax 750), at start up, the *ether* program reads from an input file which contained commands for scenario program execution with the appropriate arguments. *Ether* then starts the following programs:

> The Fire Direction Simulator (FDS), which simulates the function of the fire direction center. The FDS in turn starts the Fire Direction Officer (FDO) Display Program, which is an interface between the Fire Direction Officer and the FDS.
>
> Multiple Forward Observer Scenario (MFOSCE), which simulates the functions of target acquisition elements. There were up to four MFOSCE's.

[*] UNIX is a trademark of AT&T

[1] V.A. Kaste, G.W. Hartwig, Jr., D.C. Brodeen, C.E. Hanson, H.A. Walter, "Field Artillery Digital Message Collection and Reduction Software," Ballistic Research Laboratory Technical Report No. 2683, October 1985

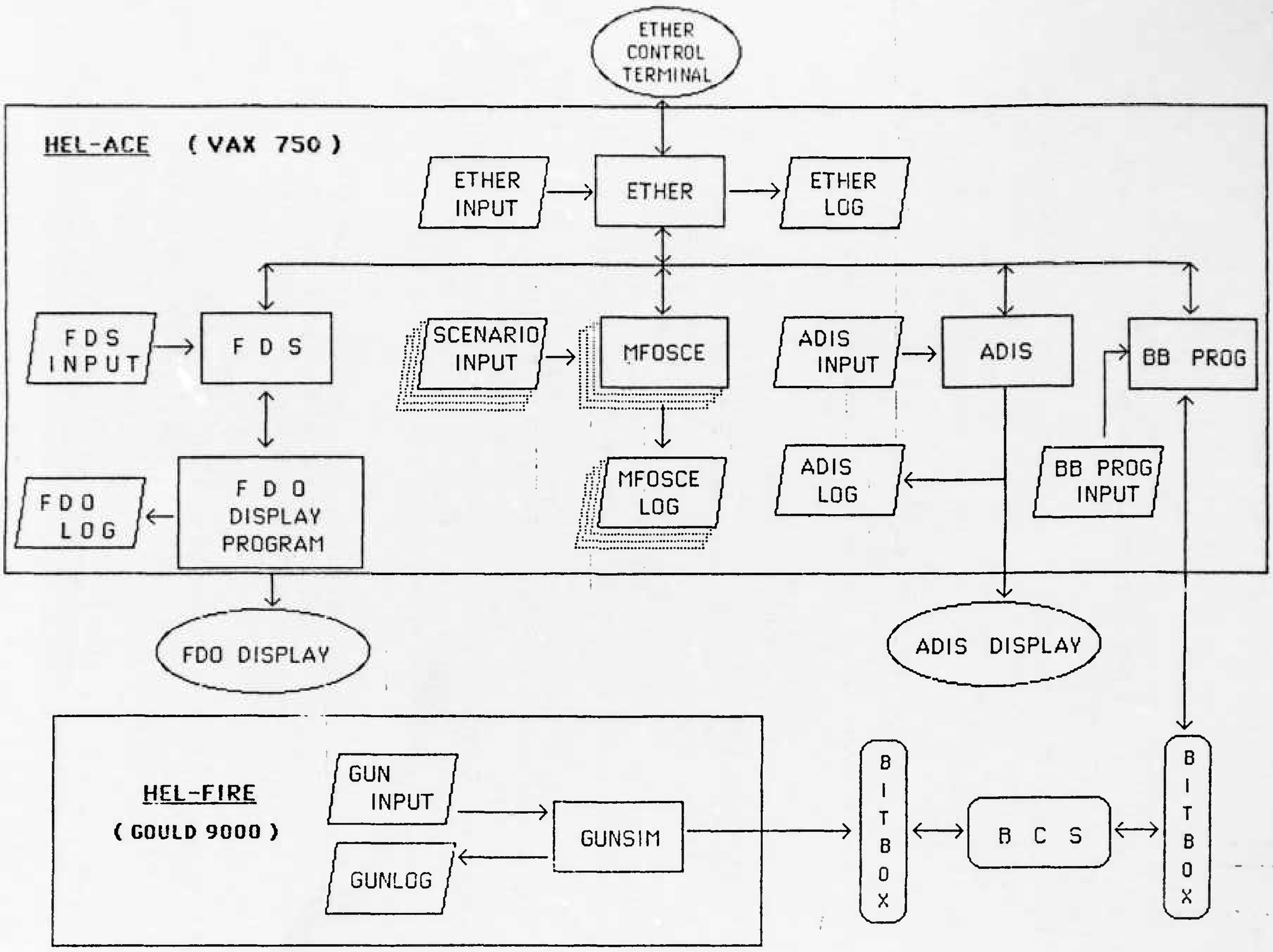

Figure 1. Firepower Control Experiment Software Configuration

The ACE Display and Information System (ADIS), a real time display of all messages that pass between ACE components during a test.

The Bit Box program, which is the interface between *ether* and a Bit Box.

At the same time *ether* is running on HEL-ACE, HEL-FIRE (a Gould 9000 computer) runs the Gun Simulator Program (GUNSIM), which simulates an artillery battery (i.e., Gun Display Units with operators and howitzer crews). The Battery Computer System[2] (BCS) communicates between the FDS and GUNSIM via Bit Boxes. Test controllers monitor the progress of the test by the means of the ADIS display.

Although the software was developed for the hardware described above, for administrative reasons, all software was run on HEL-FIRE during the experiment. This experimental fire support control software has the capability to run on either one or two computers. This is a useful feature especially when there are limited computer resources.

In this report the *ether* program as well as the modem interface program, *bb*, will be discussed.

II. ETHER

Ether is a single program written in the C programming language which functions as an intra-computer communications medium capable of supporting multiple communications networks. Experiment players are assigned to communication nets. The *ether* program manages the nets accepting messages from players and broadcasting messages to all players on the same net. *Ether* buffers incoming messages thus preventing collisions between messages on the same net.

Each *ether* network is assigned a probability of message loss which ranges from zero to one. If the probability of message loss is zero, the net is an ideal net and all messages are sent to each player on the net . If the probability of message loss is greater than zero, a uniform random number generator is used to decide whether or not a message is lost. Lost messages are not transmitted to any port on the net. Acknowledgements (acks) are treated the same as any other message.

Ether maintains a log file in which is stored a time-tagged copy of each message it receives. In addition to the raw message, each entry contains the times (Julian day, hour, minute, second) for the start of the message and the end of the message.

[2] "Cannon Battery Computer System", Computer Group, Gun Direction OL-200/GYK-29(V) (part of Computer System, Gun Direction AN/GYK-29(V)), TM 11-7440-283-12-1-1, 5 April 1984.

Although an earlier version of *ether* was able to obtain the time at which the preamble for incoming messages ended, this time is no longer available since the latest version of the Bit Boxes do not transmit the preamble to the computer. A net identification number is now encoded in this time field. This field has ceased to be useful since the latest version of the Bit Boxes do not transmit the preamble to the the computer. Messages which are "lost" are logged with a zero end of message time.

III. ETHER DESIGN

Definition of terms

The communications modeled by the *"ether"* program consists of several communications circuits, each of which can be thought of as a link; e.g. radio, wire, laser, fiber optic, etc. There are any number of communications nodes on each circuit, and each node can connect to zero or more circuits. In the UNIX environment each node represents a PROCESS#, or set of co-operating processes. Each connection between a node and a circuit involves plugging the circuit into a "socket" on the node. Because UNIX pipes* are uni-directional, a socket consists of a pair of pipes.

Summary: An "ether circuit" is a simulation of a real communications medium. It can have an arbitrary number of transceivers sharing the circuit. Each transceiver is considered to be an "ether socket".

Data Structures

Ether is based on a few key data structures. Those of primary importance are described below.

Environment structure

The first structure to be discussed is the environment structure. One "env" element is created for each node specified in the *ether* input file. This information is retained throughout the simulation for printing user-oriented diagnostics and statistics. An example of this structure is shown below:

In the UNIX environment a process is a program which is currently being executed.

* UNIX pipes are interprocess communication mechanisms which allow the exchange of data between cooperating processes.

```
struct      env {

    char e_prog[80];                 /* Simulator proc.  NULL=>empty          */
    char e_arg[256];                 /* Arg for process.  Typ term type       */
    char e_line[80];                 /* Terminal line for process             */
    char e_circuits[USOCKETS];       /* Circuit #s for sockets 0, 1, ...      */
    int  e_pid;                      /* Process id                            */
    int  e_special;                  /* forward messages to Bit Box or not?   */

};
```

Message structure

For each socket, a message structure exists which is used by the message processing subroutine to buffer an incoming message. This structure contains the arrival times, the net it arrived on and a buffer to hold partial messages.

```
struct smsg               {

        long ms_time;      /* First msg char time (Preamble/SYN char)   */
        long me_time;      /* Msg end time (0: msg lost)                */
        long sc_time;      /* Scenario time                             */
        int  m_net;        /* Channel number of msg                     */
        int  m_prelen;     /* Length of Pre-amble                       */
        int  m_len;        /* No. of chars in current message           */
        char m_buf[2048];  /* Message buffer                            */
                           /* NOTE: First char stored in m_buf[1]       */

};
```

Circuit structure

For each circuit, there are several File Descriptors (FDs) to poll. For each character received, there are several places to send it. This structure defines all the inputs and all the outputs for each ether circuit. C_state[0], c_input[0], and c_output[0] gives the state and Input/Output FDs for the 0th connection to this circuit.

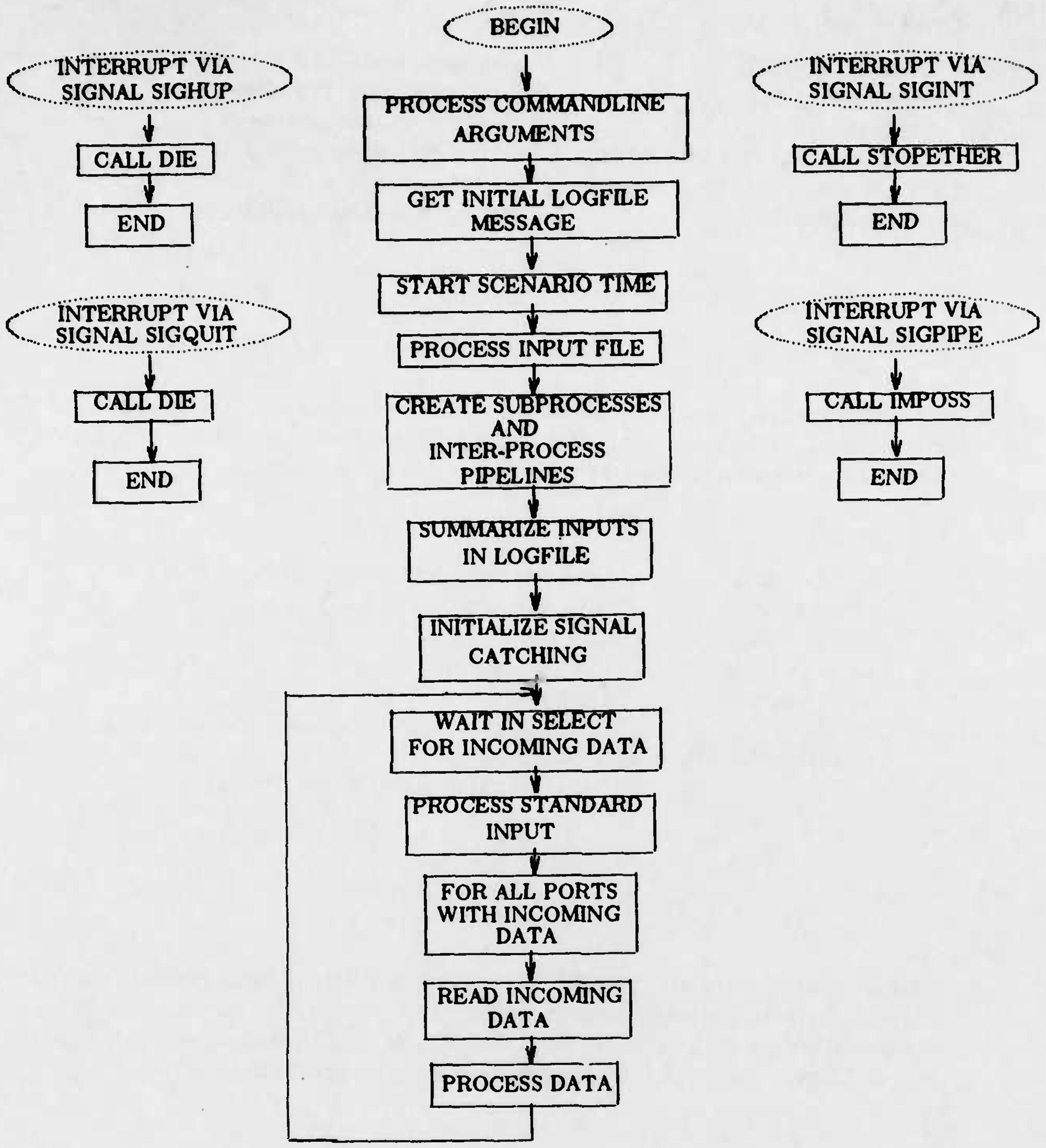
BEGIN
PROCESS COMMANDLINE ARGUMENTS
GET INITIAL LOGFILE MESSAGE
START SCENARIO TIME
PROCESS INPUT FILE
CREATE SUBPROCESSES AND INTER-PROCESS PIPELINES
SUMMARIZE INPUTS IN LOGFILE
INITIALIZE SIGNAL CATCHING
WAIT IN SELECT FOR INCOMING DATA
PROCESS STANDARD INPUT
FOR ALL PORTS WITH INCOMING DATA
READ INCOMING DATA
PROCESS DATA
INTERRUPT VIA SIGNAL SIGHUP
CALL DIE
END
INTERRUPT VIA SIGNAL SIGQUIT
CALL DIE
END
INTERRUPT VIA SIGNAL SIGINT
CALL STOPETHER
END
INTERRUPT VIA SIGNAL SIGPIPE
CALL IMPOSS
END

FIGURE 2. ETHER FUNCTIONAL DESCRIPTION

```
struct              circuits {
   int  c_state[CSOCKETS];           /* State variable for FSA recognizer     */
   int  c_input[CSOCKETS];           /* Input FD for socket, 0=> inactive     */
   int  c_output[CSOCKETS];          /* Output FD for socket                  */
   struct env *c_envp[CSOCKETS];     /* Pointer to relevant env entry         */
   long c_icount[CSOCKETS];          /* Input character count                 */
   long c_ocount[CSOCKETS];          /* Output character count                */
   int  c_mcount[CSOCKETS];          /* Msg count                             */
   int  c_lcount[CSOCKETS];          /* Lost msg count                        */
   int  c_used;                      /* # of sockets used (above)             */
   struct smsg c_msg[CSOCKETS];      /* Message data                          */
   float probt;                      /* Prob. of successful transmission      */
} circuits[NCIRCUITS];
```

Functional Description

Ether is the initial program started during a Fire Support Control experiment. BUILD_ENV, a subroutine, processes the command-line arguments, and then processes the input file storing the data in an array of structures of type "env". MAKE_PROCS is then called to do the actual process creation as well as create the pipes necessary to implement the specified networks. The resulting information is stored in an array of structures named "circuits". At this point arrangements are made to catch signals and start the experiment. The program waits in input from either the controlling terminal or one of the experiment players is detected. If there is input from the controlling terminal, it is processed first, otherwise all input ports are scanned for data. Once player input is received the FSA routine is called to process it.

FSA is an *ether* subroutine which implements a finite state automaton which extracts the message from the incoming data. Since the characters coming into the *ether* program on a player channel can be classified into a limited number of categories: namely NOISE, PREAMBLE, SYNC, MESSAGE or EOT, it is only natural to use a finite state machine to process them. Preamble characters are generated by the hardware bit box and passed on by the bit box interface program, with translation if required. Note that if the hardware does not originate the preamble characters, the simulation will not be adversely effected, but all preamble times will be zero. The SYNC part of the message is composed of the four characters <SYN><SYN><SYN><SI>. MESSAGE characters can be any ASCII character. The message is denoted by the characters that surround it and not by content. Finally the message is terminated by a string of at least four <EOT> characters and as many more as are required to complete a 16 character block. The NOISE state is characterized by all characters which appear before the SYNC.

Since it is possible to receive incomplete messages, there is a "circuits" structure maintained for each network. This structure has entries for each socket, each of which contains a buffer for holding incoming data until a complete message is obtained, as well as information necessary to restart the finite state machine properly for this socket.

Once a complete message is gathered on any socket the routine SNDMSG is called. In SNDMSG a random number is generated and compared with the probability of message loss. Depending on the results of the comparison the message is either trashed or sent to all other players on the same circuit. After the call to SNDMSG, the c_state entry in the circuits structure is set to the NOISE state and the process begins again.

All incoming messages are logged in the logfile along with the times of their arrival.

IV. ETHER PROGRAM MODULES

The functions described below are specific to the *ether* program. Functions not found here are either in the COMMON library or C library routines. In the following descriptions function names are in italics and global variable names and argument names are quoted.

(void) flags(argc, argv)

```
int argc
char **argv
```

Flags processes the command line arguments to *ether*, setting flags and file names where required. Default names are assigned for the logfile, debug outputs and input file if none are specified on the command line. The defaults are "log.ether", "bug.ether", and "input.ether", respectively.

(void) log_header()

Log_header prints an informative header in the log file including such information as descriptive log name, input file name and a time tag.

(void) u_note()

U_note prompts for and reads a message from the controlling terminal and stores it in the log file.

(void) build_env()

Build_env reads the *ether* input file and stores environment data in a structure of type "env" and circuit data in structures of type "circuit". A typical input file consists of entries of the form

```
<program> <argument list>
<I/O dev> <net number for skt #1> <net number for skt #2> ...

PR                    .2                        1
PR                    .1                        2

#                     This is a comment
```

Where <program> is the program to be invoked, <argument list> is composed of the command line arguments to be passed to the program, <I/O dev> is the terminal port which the program is to have its standard input and output connected to, and <net number for socket #n> is a simulation net which the program is to be connected to. Additional types of input lines are (1) probability of message loss, and (2) comments. A comment line is any line that begins with a '#'. Probability of loss lines consist of three fields: 1) the leading characters "PR", 2) a number between 0 and 1 indicating the probability of message loss and 3) the circuit to which this probability applies. The "circuit" array is also built by *build_env* as it processes the input file.

(void) fsa(cnum, csock, len)

int cnum, csock, len

Fsa is a finite state automaton for processing incoming messages. It is based on the fact that incoming data must consist of either noise, preamble, synchronization characters, message text or end of transmission characters. *Fsa* is called for each socket that has input data pending. It parses the incoming characters until a complete message[*] has been received and then passes the message on to *sndmsg*. Finally the "circuit" structure is reinitialized and the control is returned to the calling program. If incoming characters have all been scanned and the message is still incomplete, a return to the calling program is performed. Messages that are incomplete when a new synchronization sequence is encountered are discarded.

(char **) parse(s)

char *s

[*] Both "fixed" and "variable" format TACFIRE messages are comprised of a preamble (of variable length), the characters <SYN><SYN><SYN><SI>, the message (which includes header information) and ends with at least four <EOT> characters and as many more as are required to fill out a 16 character message block. See ref. 3 for a fuller discussion of TACFIRE messages.

This function takes a null terminated string, which contains a number of items separated by spaces and/or tabs, and returns a pointer to an array of pointers which in turn point to the individual items. Each item is null terminated in place (i.e., the original string is modified).

(void) make_procs()

Make_procs, using the information contained in the "env" array, performs the actual creation of the processes which constitute the current experiment. A process is created by a FORK system call; pipes or communication channels are formed connecting this process to the *ether* program. Finally the required program is EXECed with the appropriate argument list.

(void) summary()

Summary writes a summary of message traffic to the logfile ordered by nets.

(void) die(string)

char *string

Die is called when a catastrophic failure such as SIGHUP or SIGQUIT is detected. A message is printed on the controlling terminal and the simulation is terminated.

(void) u_menu()

U_menu processes input from the controlling terminal. Recognized commands are

'd' -	redraw the display screen
'p' -	print current probabilities of message loss and allow for input of new probabilities.
's' -	stop the *ether* after asking for confirmation
't' -	transmit a message originating from the *ether* controller
'?' -	print a menu of possible options

(char *) strcpy(out, in)

char *out, *in

Strcpy copies the string pointed to by "in" to the buffer pointed to by "out". Note that the buffer pointed to by "out" must be declared in the calling program. A pointer to the terminating null character is returned.

(void) stopether()

Stopether gracefully stops execution of the *ether* program closing logfiles and killing off child processes. The experiment controller causes this routine to be called by 's' or a SIGINT (^C) on the controlling terminal.

(void) killprocs()

Killprocs using information from the "env" structure array, kills each process of the simulation that is still running. A summary of these processes is printed in the simulation log file.

(void) imposs()

Imposs gracefully stops the simulation when a child processes dies unexpectedly. It performs the same tasks as *stopether.*

(void) stinit(cnum, csock)

int cnum, csock

Stinit initializes the appropriate "circuits" structure to prepare for reception of a message. This initialization primarily consists of setting the proper state for the *fsa* subroutine.

(void) sndmsg(cnum, csock)

int cnum, csock

Sndmsg transmits a message from socket "csock" on circuit "cnum" to all other sockets on the same circuit and to the display program. Whether a message is lost or not is determined in this routine. A random number between zero and one is generated and compared to the probability of message loss. Depending on the results of this comparison the message is either sent or not. See also *esndmsg* for further information on controller entered messages.

(int) prtstate(cnum, csock, cstate, msgp)

int cnum, csock, cstate
char * msgp

Prtstate prints the state of the *fsa* subroutine for the circuit and socket specified by "cnum" and "csocket". This includes any message currently being processed.

(void) prtmsg(csp)

struct smsg *csp

Prtmsg prints the message pointed to by "csp". Control characters are translated into '^X' notation.

(void) ptime()

Ptime converts timing information about the simulation run into seconds of elapsed time and prints the results in the log file. This information includes process time, time used in system calls, page faults, swaps and block reads and writes.

(double) ran()

This function returns a random number uniformly distributed between 0 and 1. It uses the RAND function and must be initialized by a call to the function SRAND before any call to *ran* occurs.

main(argc, argv)

int argc
char **argv

Main is the top level module of the *ether* program. It performs the initialization and calls the routines necessary to process the input file, create the log file, and start timing information. *Main* also contains the main loop for the program in which all open file descriptors are checked for pending input via a SELECT call. If data are available on the standard input, the *u_menu* module is called; otherwise, the data are read and the *fsa* routine is invoked. This process continues until the *ether* program is terminated by the controller.

(void) redrawdisplay()

Redrawdisplay sends a SIGQUIT to the display program (ADIS) which causes the display to be redrawn. This is usually done in response to a request by the experiment controller.

(void) esndmsg()

Esndmsg allows the experiment controller to send a message to a particular player from (apparently) any other player. The routine prompts the controller for header information such as to addresses and then for the message text. All messages are recorded in the log file. The message is sent to all players on the net.

(char *) find(pnt, c)

```
char *pnt
char c
```

Find returns a pointer to the next occurrence of the specified character "c" in the string pointed to by "pnt". A pointer to the null at the end of the string is returned if no occurrence of the character "c" can be found.

(void) scream(func, message, fd, circ, sock)

```
char *func
char *message
int fd, circ, sock
```

Scream is used for diagnostic prints. It prints the name of the routine in which the error was detected, the program name, and the circuit and socket information.

V. INTERPROCESS COMMUNICATION

File descriptor layout for nodes

Each node on the communications network will have pre-assigned meanings for each of the file descriptors initially passed. When invoked, the process will be given (in argv[1]) an ascii string indicating the number of active sockets. From this, relevant File Descriptors can be inferred.

F.D.	Function
0	"console" input channel
1	"console" output channel
2	diagnostic output -- goes to *ether* control tty
3	Ether Socket #1 input leg (you read this one)
4	Ether Socket #1 output leg (you write on this one)
5	Ether Socket #2 input
6	Ether Socket #2 output
:	...

Signalling mechanism

For those programs which spend most of their time waiting for input most of the time, a SIGEMT will be sent by *ether* when characters are ready to be read. When sub-programs are initiated, SIGEMT will be set to IGNORE, so programs that wish to receive SIGEMTs have to issue a

```
extern int interrupt_routine();
signal( SIGEMT, &interrupt_routine );
```

sys-call to activate the signal catching mechanism.

Limitations

With a limit of 62 open files configured at the present, it appears that a maximum of 29 connections between simulator parts and the *ether* program are possible. This should not be too restrictive. In most cases the processor speed, not the number of available file descriptors, will limit the size of simulations.

VI. DISPLAY AND CONTROL

Ether offers the experiment controller the ability to alter the probability of message loss, gracefully terminate the experiment, transmit a freetext message to a player on a particular circuit (note that all players on that circuit as well as the display program will receive copies.), and redraw the ADIS display. This is done via the simple menu shown below:

p	alter the probability of message loss
s	gracefully terminate the experiment
t	transmit a freetext message to a player on a particular circuit
d	redraw the ADIS display
?	print menu

VII. BIT BOX PROGRAM

The Bit Box program (*bb*) is a computer program that, along with a hardware device known as a Bit Box, allows the attachment of TACFIRE hardware to commercial computers. The Bit Box device is a bi-directional modem which translates the analog signals, which are used by TACFIRE for communications, into digital signals (RS232) compatible with commercial digital computers; it also translates digital signals into TACFIRE analog signals. The *bb* program listens on the terminal port to which the Bit Box is attached and serves as the interface to the ACE simulation. *bb* communicates with *ether* via pipes created by *ether* when it creates the *bb* process.

VIII. BIT BOX PROGRAM DESIGN

The Bit Box program is designed to be a two way channel for the passage of messages between a Bit Box and the *ether* program. The methodology used to accomplish this dual task is as follows. For messages from the Bit Box to the *ether*, the program blocks on a read while waiting for incoming data. When such data arrive, the program reads the data and then immediately pipes the data to the *ether*. (See Figure 3.) Note that these data[*] may or may not comprise an entire message.

Since the *ether* sends messages to the participating processes by writing the data on the pipe and then sending a signal (SIGEMT) to the target process, the *ether* to Bit Box data transfer is handled totally by an interrupt routine. This procedure, called when the SIGEMT is received, reads the data written by *ether* and writes it to the Bit Box. It continues this read-write loop until there is no longer any data on the input pipe. It then returns to whatever the program was doing when the signal was received. Since the signal processing may interrupt system calls in progress, it is necessary to check the return value from the blocking read statement to see if the read returned because data had been read or because the read was interrupted by the arrival of the signal.

IX. SOFTWARE CHANGES FOR THE FIRE POWER CONTROL TEST

Conditions of the Fire Power Control Test required that the *ether* program be modified to allow the successful completion of the test. The only modification of significance was the inclusion of code to limit the transmission of messages to the Bit Box which was mandated by the large number of messages that were not related to the scenario. Most of these messages were created only to elicit additional responses from the Fire Direction Officer with the remaining messages being special messages

[*] The only data processing performed by *bb* program is the conversion of '|' characters into bell characters <^G>s. This processing is an anachronism in that such character processing was designed for the original Bit Boxs designed and built by BRL. The '|' character corresponded to the message preamble. Since the Magnavox Bit Box do not pass the preamble to the computer, this character processing is no longer needed and could be removed from the program.

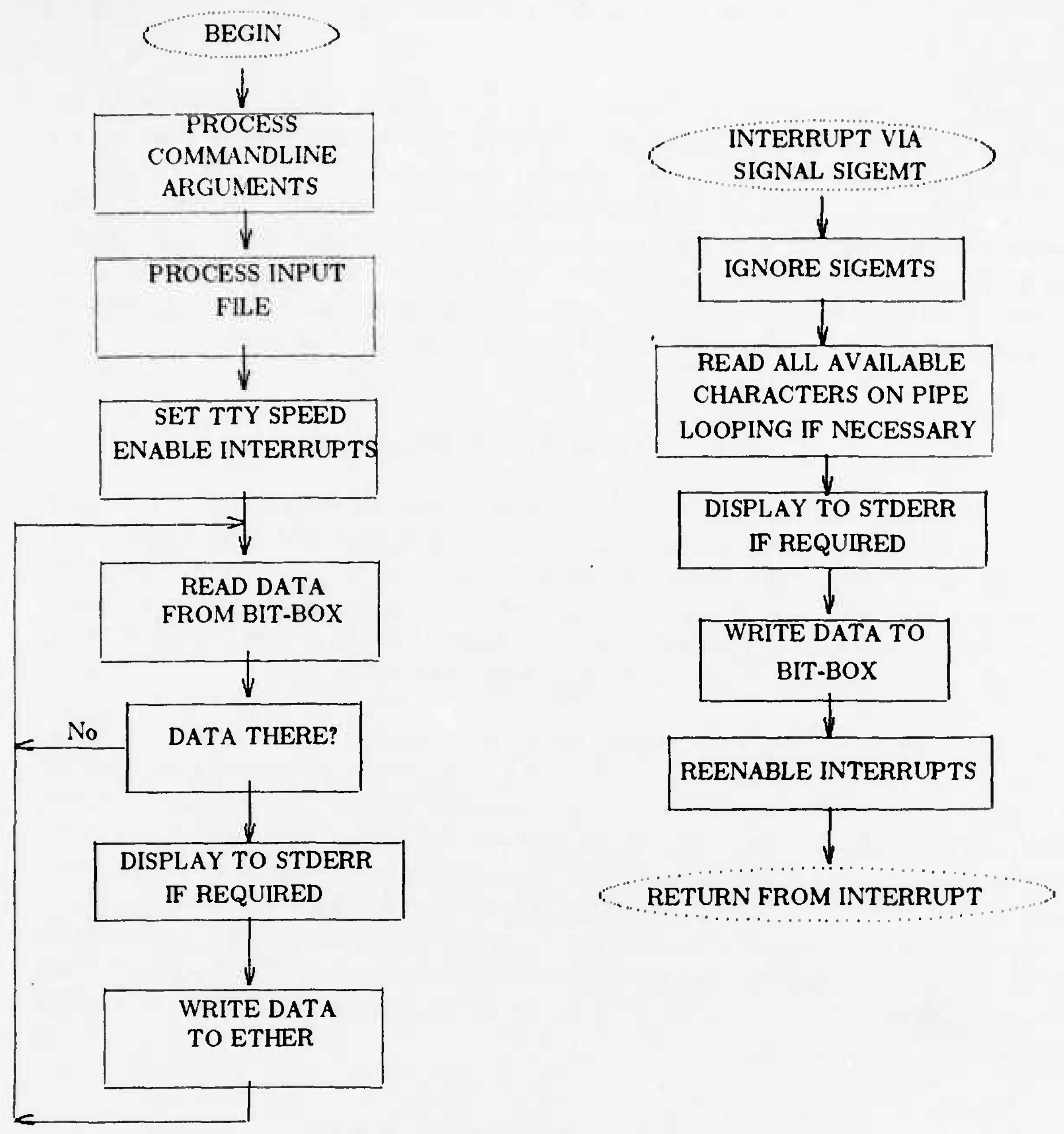
BEGIN
PROCESS COMMANDLINE ARGUMENTS
PROCESS INPUT FILE
SET TTY SPEED ENABLE INTERRUPTS
READ DATA FROM BIT-BOX
No
DATA THERE?
DISPLAY TO STDERR IF REQUIRED
WRITE DATA TO ETHER
INTERRUPT VIA SIGNAL SIGEMT
IGNORE SIGEMTS
READ ALL AVAILABLE CHARACTERS ON PIPE LOOPING IF NECESSARY
DISPLAY TO STDERR IF REQUIRED
WRITE DATA TO BIT-BOX
REENABLE INTERRUPTS
RETURN FROM INTERRUPT

FIGURE 3

BIT BOX PROGRAM FUNCTIONAL DESCRIPTION

to the *ether* for logging purposes only. Unfortunately the fact that the *ether* posted these messages to all players on the same circuit resulted in many collisions in the Bit Box device. So the decision was made to have the *ether* program not send any of these special messages to the Bit Box. This goal was accomplished by modifications to the "env" structure, the BUILD_ENV and SNDMSG functions. By sending only messages intended for players attached to the Bit Box, the problem of excessive collisions was avoided.

X. CONCLUSIONS

Ether has provided the mechanism for examining the functioning of both tactical hardware and the man - machine interface in a controlled environment. By providing for experimental controls and the accurate collection of message traffic data, *ether* has helped identify both existing and potential problems in the world of artillery command and control.

The *ether* program could use improvement in several areas. Initially a rewrite of the code to integrate the many patches made to the program would help in readability and maintainability. *Ether* should also be able to handle message traffic on a byte-by-byte basis. The current orientation (which gathers an entire message before processing) prevents the controller from having the ability to insert noise into a message. The handling of messages in their entirety also prevents collisions between messages from taking place within *ether* as in real life.

Ether should allow ACE simulations to take place on multiple machines. This would make possible much larger simulations.

REFERENCES

1. V.A. Kaste, G.W. Hartwig, Jr., D.C. Brodeen, C.E. Hanson, H.A. Walter, Jr., "Field Artillery Digital Message Collection and Reduction Software," Ballistic Research Laboratory Technical Report No. 2683, October 1985

2. "Cannon Battery Computer System", Computer Group, Gun Direction OL-200/GYK-29(V) (part of Computer System, Gun Direction AN/GYK-29(V)), TM 11-7440-283-12-1-1, 5 April 1984.

3. "Computer Program Development Specification for Computer System, Gun Direction AN/GYK-29" w/ Version 5.0 Update, EL-CG-2684A-TF, 1 Sep 1980.

BIBLIOGRAPHY

1. J. Smith, J. Grynovicki, V. Kaste, A. McKaig, S. Chamberlain, H. Walter, H. Dempsey, D. Kuhl, P. Grazaitis, "Fire Support Team Experiment," Ballistic Research Laboratory Memorandum Report No. 3422, December 1984.

2. B.W. Kernighan, D.M. Ritchie, "The C Programming Language," Prentice-Hall, New Jersey, 1978.

3. UNIX Programmer's Manual Volume 2c -- Supplementary Documents 4.2 Berkeley Software Distribution, Virtual VAX - 11 Version.

4. "Technical Report of the Training Requirements Development Facility at the US Army Field Artillery School", AAI Corporation, Report Number R604998.00002, Dec 1984.

GLOSSARY

ACE	Artillery Control Environment
ACK	Acknowledgement (message)
ADIS	ACE Display & Information System Program
ADP	Automatic Data Processing
BCS	Battery Computer System
Bit Box	Special Modem which Interfaces TACFIRE Hardware to Commercial Computers
BRL	US Army Ballistic Research Laboratory
BSD	Berkeley Software Distribution
DMD	Digital Message Device
Ether	An intra-computer communications media
FDO	Fire Direction Officer
FDS	Fire Direction Simulator
FF	Fixed Format
FR GRID	Fire Request (using Grid Coordinates for a Target Location)
GUNSIM	Gun Simulator
HEL	US Army Human Engineering Laboratory
HELBAT	HEL Battalion Artillery Test
HEL-ACE	A Digital Equipment Corporation Vax 750
HEL-FIRE	A Gould 9000 Computer
MFOSCE	Multiple Forward Observer Scenario
MTO	Message to Observer (message)
msg	message
SI	ASCII Character Hexadecimal #0F
SYN	ASCII Character Hexadecimal #16
TACFIRE	Tactical Fire Direction System
VF	Variable Format

APPENDIX A. *ETHER* Manual Page

NAME

ether – ACE communications channel simulator program.

SYNOPSIS

ether [*options*]

DESCRIPTION

Ether is the communications medium used to tie together the players in an Artillery Control Environment (ACE) scenario. The players may be simulators themselves or actual communications hardware.

Players are connected via communications channels (or *circuits)* created by the ether according to the information contained in the ether description (input) file. Players may communicate on multiple circuits.

The options are as follows:

-i The description file for this run is the next argument or 'input.ether' if one is not given. Note: If this option is not used the ether will read its description information from the standard input (this should be the terminal), and will give the user the opportunity to enter information for inclusion in the ether log file (see -l below).

-l Allows the specification of the ether log file. The filename should be given as next argument. It should be noted that the ether creates a log file regardless of whether this option is used or not. In the absence of this option or if a filename is not specified, it will use the file 'log.ether'.

-m Diagnostic current state messages to be written to the terminal when a message received by the ether is in error (message too large (overflow), invalid format (the latest character(s) received are not valid for the current state), etc.). If -d has been specified, each state of every message processed will be written. The state messages will also be written to the debug output file if -D has been specified. For information regarding debug output, see options -d and -D.

-D Information relative to program debugging is written to the file specified as next argument, or to the file 'bug.ether' if a filename is not given.

-d Information relative to program debugging is written to the Unix standard error output. This is normally the ether control terminal (the terminal from which the ether was started).

DESCRIPTION FILE FORMAT

The description file consists of lines of text, specifying circuit message loss probabilities, player processes and their arguments, control terminals, and circuit numbers. The lines are parsed according to the criteria below.

Lines beginning with "#" are comments and are ignored.

Lines beginning with "PR" are considered to contain the probability of message loss for a specific circuit. Two arguments separated by spaces should follow the PR, the first stating the probability and the second a circuit number. The probability is a number between 0 and 1. A probability of .5 would cause a random loss of half of all messages on a circuit.

All other lines are considered to describe player processes, the control terminals and circuits to which they are connected. These descriptions consist of two lines each. The first line contains the player process and its arguments separated by spaces. Characters usually considered to have special meaning to the Unix shells are NOT expanded or recognized. However, the standard input of the process may be modified using '<' as in sh(1).

The second line contains a set of arguments, the first of which designates the name of the control terminal to which the standard input and output of the process (file descriptors 0 and 1) will be attached. The standard error output of the process (file descriptor 2) remains attached to that of the ether. Remaining arguments are taken to be the circuit numbers to which the process will be connected.

Lines beginning with "$" describe one special process which is to receive all messages sent on all circuits. Its description matches the two line format of the other process descriptions with the exception that there are no circuit numbers to be specified on line 2. There can only be one such process.

EXAMPLE

A sample description file follows:

```
#       Probabilities of message loss...
PR   .2   1
PR   .2   2

#       The fire direction simulator...
fds 6 < input.fds
/dev/tty 1

#       The adis display program...
$/d/ace/.bin/adis -ali input.adis
$/dev/tty16

#       The forward observers...
/d/ace/.bin/mfosce 1 1 -Dli input.fosce
/dev/null 2

/d/ace/.bin/mfosce 2 2 -Dli input.fosce
/dev/null 2

#       The fist connections...
/d/ace/.bin/bb
/dev/tty21 1

/d/ace/.bin/bb
/dev/tty22 2
```

SEE ALSO

fds(1), mfosce(1), adis(1), bb(1)

BUGS

The communications scheme of the ether depends on the maintenance of i/o channels, called pipes, between the parent ether process and its descendant processes. The pipes consume a limited per-process resource known as a *file descriptor*. This necessarily relates the maximum number of descendant processes possible in a scenario to the number of file descriptors available. There are three file descriptors used by the ether itself and an additional two are required for each descendant's pipes. Therefore: max descendants = (max file descriptors available - 3) / 2. The maximum number of file descriptors available per-process on the system is defined in /usr/include/sys/param.h as the pre-processor constant 'NOFILE'.

APPENDIX B. *bb* Manual Page

NAME

bb – ACE Bit-Box driver program.

SYNOPSIS

bb [-f] [-i *inputfile* **] [-t]**

DESCRIPTION

Bb is the communications driver program used to attach a Digital Message Device (DMD) to an Artillery Control Environment (ACE) simulation. A hardware device known as a *bit-box* serves as the hardware interface between the DMD and the computer, and is the means through which bb establishes communication between the DMD and the other players in an ACE simulation.

Specifically, bb listens and transmits on i/o channels supplied to it by an *ether(1)* and defined in an *ether description file*. Those are: the channel between bb and the bit-box, and the channel between bb and an *ether circuit*. The ether creates the communications environment necessary for the transmission of messages between bb and the players in the simulation via the circuit, and connects bb to the Unix i/o port connected to the bit-box device. The i/o port name and circuit number are specified in the description file on line 2 of the bb player process description. See ether(1).

The options are as follows:

- **-f** Messages sent *from* bb to the ether will be printed on the standard error output.
- **-i** The filename given as next argument will be read as input. Specifications concerning the input file are discussed below.
- **-t** Messages sent *to* bb from the ether will be printed on the standard error output.

INPUT FILE FORMAT

The input file contains lines of text of three forms and is parsed according to the criteria below.

A line beginning with the characters "bb" is considered to contain a baud rate specification for the i/o line between bb and the bit-box. Following the "bb" there may be spaces and/or tabs and one of the following baud rates specifications:

110, 300, 600, 1200, 2400, 4800, or 9600.

The baud rate of the line is changed immediately. At program startup, the assumed baud rate is 1200.

A line beginning with the character "\" and containing a terminating ">" is passed immediately to the bit-box as a bit-box command. The content of the command is dependent upon the protocol of the bit-box hardware device.

Lines beginning with "#" are comments and are ignored, as are blank lines.

FILES

Bb file descriptors 0 and 1 are attached by the ether to the Unix i/o port (/dev/tty??).

Bb file descriptors 3 and 4 are connected to the ether itself.

The standard error output (file descriptor 2) is attached to that of the ether. This is normally the terminal from which the ether was started.

SEE ALSO

ether(1), fds(1), mfosce(1), adis(1)

BUGS

APPENDIX C. BIT BOX SPECIFICATIONS

The contractor will design and fabricate hardware, and provide software to implement a Tactical Communication Modem (TCM) that provides bidirectional RS232-FSK conversion of TACFIRE message format data. The TCM will provide the appropriate communication data formatting, signaling, electrical interface, and control protocol to send and receive TACFIRE message format data on a RS232 full duplex interface and a TACFIRE FSK net. A message received on either interface will be recovered, buffer stored if buffer space permits, and retransmitted on the other interface.

The Tactical Communication Modem will provide the operational and functional capabilities as set forth herein.

1. The TCM Frequency-Shift-Keying (FSK) modem will provide, as specified below, the proper communication data formatting, communication protocol, signaling, error detection and correction, and equipment interfaces necessary to send and receive TACFIRE FSK messages in the single and double block mode.

a. The FSK modem will send and receive messages using the TACFIRE communication format. The communication format for each message consists of three contiguous sections. The three sections consist of a variable length preamble, a 32-bit message synchronization word, and the message format data proper organized into 16-character data blocks.

(1) The preamble consists of an alternate 1-0 bit pattern variable from 0.2-4.0 + 0.1 seconds.

(2) The 32-bit four-character message synchronization word will consist of three SYN characters and one SI or NULL character. Each synchronization character consists of a seven-bit ASCII character plus one odd parity bit.

(3) The message format data will be sent and received in complete 16-character data blocks. Each of the sixteen 12-bit characters in the data block consists of seven ASCII data bits plus five hamming code parity bits. The data bits in each data block will be transmitted and received using the time dispersed format. A minimum of four EOT characters located at the end of the last data block must terminate a message transmission.

b. The TCM FSK modem will transmit and receive TACFIRE FSK messages in the single or double block mode as shown in figure 4.

c. The TCM FSK modem will provide single-bit error correction and double-bit error detection on received FSK messages. In the double block mode correct or correctable characters from both identical message blocks will be merged to form, if possible, a single correct block of data characters.

d. The TCM FSK modem will send and receive data bits using 1200 Hz (logic

1)/2400 Hz (Logic 0) continuous phase FSK signaling at 600 and 1200 baud.

e. The TCM FSK modem will provide the proper signal and impedance matching, electrical signal isolation from attached communication equipment, and the appropriate control signals to interface to standard army tactical FM radios and wireline equipment.

f. The TCM FSK modem will provide the required ACK delay net protocol. The TCM will accept an incoming FSK message if the received message buffer is available, and will respond with an ACK message if programmed to do so.

2. The TCM will provide a full duplex RS232 interface with the following characteristics and capabilities.

a. All RS232 data and control signals will conform to RS232C electrical signal and signal sense interface requirements.

b. All characters will be ASCII characters, and each character will be transmitted least significant bit first.

c. The TCM will operate as Data Communications Equipment (DCE).

d. The TCM RS232 interface will operate at any one of the following baud rates: 150, 300, 1200, 2400, 4800, or 9600.

e. The RS232 word parameters (word length, parity enable and type, and number of stop bits) and baud rate will be operator selectable via switches located on the FSK modem card.

f. Messages received by the TCM on the RS232 interface must be terminated by a minimum of four EOT characters.

3. The TCM will receive and buffer store one message from either interface.

4. Formatted control/command messages will be sent to the TCM on the RS232 interface that will allow the user to enter operational parameters such as preamble length, FSK baud rate, single/double block mode, and the message destination address (es) to be accepted by the TCM.

5. The TCM will provide chassis mounted connectors for power, RS232 interface, FSK radio, and wireline connections.

6. All parts and construction will be to best commercial practice consistent with operation in a sheltered room temperature environment.

BRL MANDATORY DISTRIBUTION LIST

No of Copies	Organization
lass., unlimited) 12 lass., limited) 2 sified) 2	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
1	HQDA (SARD-TR) Washington, DC 20310-0001
1	Commander US Army Materiel Command ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander US Army Laboratory Command ATTN: AMSLC-DL Adelphi, MD 20783-1145
2	Commander Armament RD&E Center US Army AMCCOM ATTN: SMCAR-MSI Picatinny Arsenal, NJ 07806-5000
2	Commander Armament RD&E Center US Army AMCCOM ATTN: SMCAR-TDC Picatinny Arsenal, NJ 07806-5000
1	Director Benet Weapons Laboratory Armament RD&E Center US Army AMCCOM ATTN: SMCAR-LCB-TL Watervliet, NY 12189-4050
1	Commander US Army Armament, Munitions and Chemical Command ATTN: SMCAR-ESP-L Rock Island, IL 61299-5000
1	Commander US Army Aviation Systems Command ATTN: AMSAV-DACL 4300 Goodfellow Blvd. St. Louis, MO 63120-1798
1	Director US Army Aviation Research and Technology Activity Ames Research Center Moffett Field, CA 94035-1099

No of Copies	Organization
1	Commander US Army Missile Command ATTN: AMSMI-RD-CS-R (DOC) Redstone Arsenal, AL 35898-5010
1	Commander US Army Tank Automotive Command ATTN: AMSTA-TSL (Technical Library) Warren, MI 48397-5000
1	Director US Army TRADOC Analysis Command ATTN: ATAA-SL White Sands Missile Range, NM 88002-5502
(Class. only) 1	Commandant US Army Infantry School ATTN: ATSH-CD (Security Mgr.) Fort Benning, GA 31905-5660
(Unclass. only) 1	Commandant US Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905-5660
1	AFWL/SUL Kirtland AFB, NM 87117-5800
(Class. only) 1	The Rand Corporation P.O. Box 2138 Santa Monica, CA 90401-2138
1	Air Force Armament Laboratory ATTN: AFATL/DLODL Eglin AFB, FL 32542-5000
	Aberdeen Proving Ground Dir, USAMSAA ATTN: AMXSY-D AMXSY-MP, H. Cohen Cdr, USATECOM ATTN: AMSTE-TO-F Cdr, CRDEC, AMCCOM ATTN: SMCCR-RSP-A SMCCR-MU SMCCR-MSI

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
2	Principal Dep Under Secretary of Defense for Research and Engineering The Pentagon Washington, D.C. 20301
2	Ofc of Dep Under Secretary of Defense for R&E (Research and Advanced Technology) Office of Electronics & Physical Sciences The Pentagon Washington, D.C. 20301
2	Ofc of Dep Under Secretary of Defense for R&E (Research and Advanced Technology) Office of Engineering Technology The Pentagon Washington, D.C. 20301
2	Ofc of Dep Under Secretary of Defense for R&E (Tactical Warfare Programs) The Pentagon Washington, D.C. 20301
2	Director Defense Advanced Research Projects Agency ATTN: Info Processing Techniques Office 1400 Wilson Boulevard Arlington, VA 22209
1	Director Defense Advanced Research Projects Agency ATTN: Tactical Technology Ofc 1400 Wilson Boulevard Arlington, VA 22209
4	Director Defense Advanced Research Projects Agency Information Science and Technology Office ATTN: Dr. Perry LTC Baker LTC Montie Mr. Amarel 1400 Wilson Boulevard Arlington, VA 22209
1	Ofc of Dep Under Secretary of the Army (Operations Research) The Pentagon Washington, DC 20310

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
5	HQDA ATTN: DAIM-FCL, LTC Theroux DAMA-CSC DAMO-FDQ DAMO-ZE DAMA-WSZ-A Washington, DC 20310
1	US Army Logistics Center ATTN: ATCL-SST, Mr. Thomas Fort Lee, VA 23801-6000
3	Office of Assistant Secretary of the Army (Research, Development, and Acquisition) ATTN: Deputy for Fire Support Deputy for Comm & Target Acquisition Deputy for Science and Technology The Pentagon Washington, D.C. 20310
1	HQDA (DAMA-TR)IL Washington, D.C. 20310
1	HQDA (DAMO-FDQ)IL Washington, D.C. 20310
1	HQDA (DAMO-C4)IL Washington, DC 20310
1	HQDA (DAMA-RQC)IL Washington, D.C. 20310
1	HQDA (DAMO-FDI)IL Washington, D.C. 20310
1	HQDA (DAMO-FDE)IL Washington, D.C. 20310
1	HQDA (DAMA-ZE)IL Washington, D.C. 20310
1	HQDA (DAMA-RAX)IL Washington, D.C. 20310
1	HQDA (DAMA-ARZ-A) Washington, D.C. 20310
1	HQDA (DAMA-CSC)IL Washington, DC 20310
1	HQDA (DAMA-SCS) Washington, D.C. 20310
1	HQDA (DAMA-WSZ-A) Washington, D.C. 20310
1	HQDA (DAMA-WSW)IL Washington, D.C. 20310
1	Commander US Army Materiel Command ATTN: AMCDE-SC 5001 Eisenhower Avenue Alexandria, VA 22333

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Materiel Command ATTN: AMCDE-SB 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: AMCDE-SG 5001 Eisenhower Avenue Alexandria, VA 22333
11	Commander Armament R&D Center US Army AMCCOM ATTN: SMCAR-CG SMCAR-ASIL SMCAR-LC SMCAR-LCS SMCAR-SCIL SMCAR-SCF SMCAR-SCF-AV SMCAR-SCF-CS SMCAR-TD SMCAR-TDS SMCAR-TSS Dover, NJ 07801-5001
1	Commander US Army Aviation Research and Development Command ATTN: AMSAV-E 4300 Goodfellow Boulevard St. Louis, MO 63120
1	Commander US Army Avionics Research and Development Activity ATTN: DAVAAIL Fort Monmouth, NJ 07703

No. of Copies	Organization
16	Commander US Army Communications Research and Development Command ATTN: AMSEL-RD-COM-D, (Mr. Famolari) AMSEL-RD-COM-AF, (Mr. Quigley, Mr. Kernan, Ms. Roberts) AMSEL-RD-COM-AF-1, (Mr. Salton, Mr. Levine, Mr. Kawnzinger, Mr. Polanchyck, Mr. Kamenel) AMSEL-RD-COM-AF-2, (Mr. Graff, Mr. Bereschinsky) AMSEL-RD-COM-IE-2, (Mr. Genlot) AMSEL-RD-COM-RF AMSEL-RD-COM-RN AMSEL-RD-COM-RS, (Dr.Dworkin) AMSEL-RD-COM-RX Fort Monmouth, NJ 07703-5000
7	Commander US Army Communications Research and Development Command ATTN: AMSEL-PPA-SA AMSEL-SEI AMSEL-SEI-A AMSEL-SEI-E AMSEL-SEI-I AMSEL-SEI-V AMDCO-TCS Fort Monmouth, NJ 07703
1	Commander US Army Electronics Research and Development Command Technical Support Activity ATTN: AMLET-ID Fort Monmouth, NJ 07703

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
1	Commander ERADCOM Technical Library ATTN: DELSD-L (Reports Section) Fort Monmouth, NJ 07703-5301
3	Commander Atmospheric Sciences Laboratory ATTN: AMLAS AMLAS-BEIL AMLAS-DIL White Sands Missile Range, NM 88002
3	Director Electronic Warfare Laboratory ATTN: AMLEW AMLEW-CIL AMLEW-DIL Fort Monmouth, NJ 07703
5	Commander US Army Harry Diamond Labs. ATTN: AMLHD-TD, Dr. Scully AMLHD-NW-EMB, (George Gornak) AMLHD-PPIL AMLHD-D-O AMLHDIL 2800 Powder Mill Road Adelphi, MD 20783
5	Director Night Vision & Electro-Optics Laboratory ATTN: AMLNV AMLNV-DIL AMLNV-ACI AMLNV-SEIL AMLNV-SIIL Fort Belvoir, VA 22060

No. of Copies	Organization
1	National Security Agency ATTN: S743 (Mr. Spano) 9800 Savage Rd. Fort George G. Meade, MD 20755
3	Director US Army Signals Warfare Laboratory ATTN: AMLSWIL AMLSW-RAIL AMLSW-CEIL Vint Hill Farms Station Warrenton, VA 22186
2	Commander US Army Missile Command ATTN: AMSMI-RNIL AMSMI-YDL Redstone Arsenal, AL 35898-5630
1	Commander US Army Engineer and Topographic Laboratories ATTN: ETL-TDIL Fort Belvoir, VA 22060-5603
2	Commander US Army Foreign Science and Technology Center ATTN: AMXST-IS-I AMXST-CA-I Federal Office Bldg 220 7th St. NE Charlottesville, VA 22901
1	Commander US Army Research Office Box 12211 Research Triangle Park, NC 27709-2211

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Research, Development and Standardization Group Australia APO, SF 96404
1	Department of Army Scientific and Information Team (Eur) Box 48 APO, New York 09710
1	Commander US Army Research, Development and Standardization Group United Kingdom USARDSG (UK) Box 65 FPO New York 09510
1	Project Manager, Cannon Artillery Weapon Systems US Army AMCCOM ATTN: AMCPM-CAWS Dover, NJ 07801
1	Assistant Program Manager Joint Tactical Fusion Field Office ATTN: JTF-CAC Vint Hill Farms Station Warrenton, VA 22186-5182

No. of Copies	Organization
1	Project Manager Control and Analysis Center ATTN: AMCPM-CAC Vint Hill Farms Station Warrenton, VA 22186
1	Project Manager Multiple Launch Rocket System ATTN: AMCPM-RS Redstone Arsenal, AL 35898
1	Project Manager Operations Tactical Data Systems ATTN: AMCPM-OPTADS Fort Monmouth, NJ 07703
1	Project Manager Position Location Reporting System Tactical Information Distributing System ATTN: AMCPM-PL Fort Monmouth, NJ 07703
1	Project Manager Remotely Piloted Vehicle ATTN: AMCPM-RPV 4300 Goodfellow Blvd St. Louis, MO 63120
1	Project Manager Single Channel Ground & Airborne Radio System ATTN: AMCPM-GARS Fort Monmouth, NJ 07703
3	Project Manager, TACFIRE/Field Artillery Tactical Data Sys ATTN: AMCPM-TFIL Fort Monmouth, NJ 07703

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
1	Project Manager, TACFIRE Software Support Group ATTN: AMCPM-TF-FS Fort Sill, OK 73503
2	Project Manager Training Devices ATTN: AMCPM-TND AMCPM-TND-SE Orlando, FL 32813
1	US Army Training Support Center ATTN: ATSC Fort Eustis, VA 22604
1	Chief US Army Strategy and Tactics Analysis Group 8120 Woodmont Avenue Bethesda, MD 20014
8	Commander US Army Training & Doctrine Command ATTN: ATCD ATCD-AIL ATCD-CIL ATIC-NC ATCD-EIL ATCD-FIL ATTEIL ATDOIL Fort Monroe, VA 23651
1	Commander HQ USACAC & FT LVN ATTN: ATOR-C Fort Levenworth, KS 66027-5080
5	Commander HQ USACAC & FT LVN ATTN: ATZL-CA ATZL-CAD-L ATZL-CAC-I ATZL-CAC-CC, (CPT Greene) ATZL-CAM-I Fort Levenworth, KS 66027-5080
2	Director US Army TRADOC Systems Analysis Activity ATTN: ATOR-T ATOR-T-SL White Sands Missile Range, NM 88002-5502
1	Director US Army TRADOC Systems Analysis Activity ATTN: ATOR White Sands Missile Range, NM 88002-5502
2	Commander US Army Combat Developments Experimentation Center ATTN: ATEC ATEC-LRPP-TPD Fort Ord, CA 93941
7	President US Army Field Artillery Board ATTN: ATZR-BD ATZR-BDCT ATZR-BDWT ATZR-BDAS HEL Liaison Officer (3 cys) Fort Sill, OK 73503

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
1	Commandant US Army Armor School ATTN: ATSB-CD Fort Knox, KY 40121
5	Commander US Army Aviation Center ATTN: ATZQ (2 cys) ATZQ-TSM-A ATZQ-TSM-S ATZQ-TSM-H Fort Rucker, AL 36360
1	Commander US Army Field Artillery School ATTN: ATZR-C Fort Sill, OK 73503
6	Commander US Army Field Artillery School ATTN: ATSF-AIL ATSF-DIL ATSF-DUS ATSF-E ATSF-F ATSF-TIL Fort Sill, OK 73503
1	Commander US Army Field Artillery School ATTN: ATSF-GIL Fort Sill, OK 73503
4	Commander US Army Field Artillery School ATTN: ATSF ATSF-SD ATSF-T ATSF-WIL Fort Sill, OK 73503

No. of Copies	Organization
4	Commander US Army Field Artillery School ATTN: ATSF-C ATSF-C/Materiel ATSF-C/Concepts ATSF-C/Systems Fort Sill, OK 73503
4	Commander US Army Field Artillery School ATTN: ATSF-C/Analysis ATSF-C/Data Sys ATSF-CD ATSF-CT (E. Stiles) Fort Sill, OK 73503
4	Commander US Army Field Artillery School ATTN: ATSF-TSM-TF ATSF-TSM-TD ATSF-TSM-C ATSF-TSM-CN Fort Sill, OK 73503
5	Commander US Army Field Artillery School ATTN: ATSF-TSM-FF ATSF-TSM-MR ATSF-TSM-MLRS ATSF-TSM-PE ATSF-TSM-RPV Fort Sill, OK 73503
3	Commandant US Army Intelligence Center and School ATTN: ATSI ATSI-CD-AS, (LT Ochner, CPT Simon) Fort Huachuca, AZ 85613

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
10	Commander
	US Army Signal Center & School
	ATTN: ATZH
	ATZH-AD
	ATZH-ATIL
	ATZH-CD (3 cys)
	ATZH-CDC
	ATZH-CDT,(CPT Travis)
	ATZH-CDM,(Mr. Grissom)
	ATZH-SGIL
	Fort Gordon, GA 30905
3	Commandant
	US Army Intelligence Center and School
	ATTN: ATSI
	ATSI-CD (2 cys)
	Fort Huachuca, AZ 85613
1	Commandant
	US Army Command and General Staff College
	Fort Levenworth, KS 66027
1	Commandant
	US Military Academy
	West Point, NY 10996
1	US Naval Academy
	Annapolis, MD 21404
1	US Air Force Academy
	Colorado Springs, CO 80840
1	Commander
	US Army Operational Test and Evaluation Agency
	5600 Columbia Pike
	Falls Church, VA 22041
1	Commander
	Naval Surface Weapons Center
	ATTN: Technical Director
	Silver Spring, MD 20910
1	Commander
	Naval Surface Weapons Center
	Weapons Laboratory
	ATTN: Technical Director
	Dahlgren, VA 22448
1	Commander
	Naval Weapons Center
	China Lake, CA 93555
1	Office Chief (at Navy A.I.)
	Navy Research Laboratories
	ATTN: Jude Franklin
	Code 7510
	Washington, DC 20375
1	Commander
	XVIII Airborne Corps
	ATTN: Comm Elec Bd, ADDS Experiment
	Fort Bragg, NC 28307
3	Director
	ADDCOMPE Test Division
	ATTN: ATXA-BDD
	(LTC McCarson,
	Mr. Adams,
	Mr. Peeler)
	Army Communications Electronics Board
	Fort Bragg, NC 28307
1	Chief, Ground Operations Div
	Development Center
	Marine Corps Development and Education Command
	Quantico, VA 22134

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
3	Commander US Army Development & Employment Agency ATTN: MODE MODE-DCCS MODE-C31, (MAJ Forbes) Fort Lewis, WA 98433
1	Commander US Army Development and Employment Agency ATTN: MODE-TED-SAB Fort Lewis, WA 98433
1	Commander ATTN: DIVARTY 9th Infantry Division Fort Lewis, WA 98433
1	CO MCTSSA, MCB Camp Pendelton, CA 92055
1	AAI Corporation ATTN: John Foster P.O. Box 6767 Baltimore, MD 21204
1	Advanced Information & Decision Systems ATTN: Dr. Abram 201 San Antonio Circle Suite 286 Mountain View, CA 94040
1	AFELM, The Rand Corporation ATTN: Library-D 1700 Main Street Santa Monica, CA 90406
3	BBN Laboratories ATTN: Ms. Westcott Mr. Gunshenan Mr. Hinden 10 Moulton St. Cambridge, MA 02238
3	BBN Communications Corporation ATTN: Mr. Pogran Mr. Corini Mr. Forsdick 50 Moulton Street Cambridge, MA 02238
1	Bowen-McLaughlin York Co P.O. Box 1512 York, PA 17405
1	Computer Sciences Corporation ATTN: Al Coates Suite G 2719 Pulaski Highway Edgewood, MD 21040
5	Hazeltine Corporation ATTN: Mr. Groff Mr. Markel Mr. Willins Mr. Hunter Mr. Rosanes Cuba Hill Road Greenlawn, NY 11740
3	LB&M Associates ATTN: Tony Pokorny Joe Halloran Bob Robillard 111 SW C Ave. Suite 200 Lawton, OK 73501

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
1	Litton Data Systems ATTN: W.E. Knebel (MS 45-09) 8000 Woodley Avenue VanNuys, CA 91409
1	Litton Guidance & Control Systems ATTN: Robert W. Maughmer 5500 Canoga Avenue Woodland Hills, CA 91364
1	Lockheed Electronics Co. Inc. Systems Division ATTN: Dr. Knox 1501 US Highway 22 Plainfield, NJ 07061
6	Magnavox Electronics Systems Co. Tactical Systems ATTN: Mr. Willis Mr. Dixon Mr. Kavara Mr. Whiteman Mr. Norris Mr. Charles 1313 Production Road Fort Wayne, IN 46808
1	Norden Systems, Inc Norden Place Norwalk, CT 06856
1	SAIC Comsystems ATTN: Jan Dolphin/Niel Blank 2801 Camino Del Rio South San Diego, CA 92129
2	Science Applications, Inc. ATTN: David Erickson Richard Scaglione P.O. Box 2351 La Jolla, CA 92038

No. of Copies	Organization
5	SRI, International Advanced Information Technology Applications Dept. ATTN: EJ113 (Ms. Lee) EJ323 (Mr. Schreier) EJ329 (Mr. Hastie) EJ347 (Mr. Frankel) EJ390 (Mr. Au) 333 Ravenswood Ave. Menlo Park, CA 94025
1	SRI-Washington ATTN: Mr. Page 1611 N. Kent St. Arlington, VA 22209
3	SRI Field Office ATTN: Dr. Hagan Mr. Borchert Mr. Gorman P.O. Box 70314 Fort Bragg, NC 28307-5000
2	SRI Field Office ATTN: Mr. Kozel Mr. Holman P. O. Box 33281 Fort Lewis, WA 98433
3	System Development Corporation ATTN: Mr. Brinkely Mr. Williams Mr. Cole McLean Research Center 7929 Westpark Dr. McLean, VA 22102
1	SDC Systems Group ATTN: J. Williams 7929 Westpark Dr. McLean, VA 22102

AUTHOR'S DISTRIBUTION LIST

No. of Copies	Organization
3	TELOS Computing, Inc. P.O. Box 846 Lawton, OK 73502
1	Titan Systems Inc. ATTN: Mr. Bloom P.O. Box 2123 Chatsworth, CA 91313-2123
4	The MITRE Corporation Washington C3I Operations ATTN: Judy Dahmann Ms. Kirk Mr. Royster Mr. Perry 1820 Dolley Madison Blvd. McLean, VA 22102
1	Rockwell International Collins Communications Systems Division ATTN: Mr. Caples 3200 E. Renner Road, CS.7 MS 460-215 Richardson, TX 75081
1	Vecter Research, Inc. ATTN: Gary Witus 2536 Packard Road Ann Arbor, MI 48104
1	Central Intelligence Agency Office of Central Reference Dissemination Branch Room GE-47 HQS Washington, D.C. 20502

No. of Copies	Organization
	Aberdeen Proving Ground, MD 21005
8	Dir, USAMSAA ATTN: AMXSY-G AMXSY-GI AMXSY-GS (2 cys) AMXSY-CIL (3 cys) AMXSY-R
5	Cdr, USATECOM ATTN: AMSTE-CM-F AMSTE-AD-A AMSTE-AD-I AMSTE-AD-S AMSTE-CT-C
2	Dir, USACSTA ATTN: STECS-AS STECS-AS-L
20	Dir, USAHEL ATTN: AMXHE-D AMXHE-FT (15 cys) AMXHE-CS AMXHE-CC AMXHE-SPIL AMXHE-FS (Library)
3	Commandant, USAOC&S ATTN: ATSL-CD (3 cys)
3	C, Field Supt Div USAFSTC (3 cys)
1	PM Smoke ATTN: AMCPM-SMK

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number ____________________ Date of Report __________

2. Date Report Received ____________________

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) ____________________

4. How specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.) ____________________

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided or efficiencies achieved, etc? If so, please elaborate. ____________________

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

CURRENT ADDRESS

Name

Organization

Address

City, State, Zip

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD ADDRESS

Name

Organization

Address

City, State, Zip

(Remove this sheet, fold as indicated, staple or tape closed, and mail.)

— — — — — — — — — — — FOLD HERE — — — — — — — — — — —

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 12062 WASHINGTON,DC
POSTAGE WILL BE PAID BY DEPARTMENT OF THE ARMY

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-9989

— — — — — — — — — — — FOLD HERE — — — — — — — — — — —